GraphPad: Inference-Time 3D Scene Graph Updates for Embodied Question Answering

Anonymous CVPR submission

Paper ID 8

Abstract

Structured scene representations are a core component of embodied agents, helping to consolidate raw sensory streams into interpretable, modular, and searchable formats. Due to their high computational overhead, many approaches build such representations in advance of the task. However, when the task specifications change, such static approaches become inadequate as they may miss key objects, spatial relations, and details. We introduce ***GraphPad****, a modifiable structured memory that an agent can tailor to the needs of the task through API calls. It comprises a mutable scene graph representing the environment, a navigation log indexing frame-by-frame content, and a scratchpad for task-specific notes. Together, GraphPad serves as a dynamic workspace that remains complete, current, and aligned with the agent's immediate understanding of the scene and its task. On the OpenEQA benchmark, GraphPad attains* ***55.3 %*** *accuracy—****+3.0 pp*** *over an image-only baseline using the same vision–language model—while operating with* ***five times fewer*** *input frames. These results show that allowing online, language-driven refinement of 3-D memory yields more informative representations without extra training or data collection.*

1. Introduction

A household robot has just scanned the kitchen when a user asks, *"Is the red mug back inside the upper cupboard?"* Because the scene graph was built from only a handful of earlier keyframes, it lacks both a node for the mug and the relation linking it to the cupboard interior. With this critical information missing, the agent must either guess, rescan the entire scene, or refuse the query altogether. This scenario highlights a fundamental limitation in current embodied AI: **structured 3D memories are typically finalized before the downstream task is known, frequently omitting objects and spatial relations that later prove essential for action or reasoning**.

Inherent limitations of static memories. Modern embodied systems commonly pair a vision–language model (VLM) with a pre-computed 3D scene graph or semantic map for symbolic reasoning [11, 12]. These graphs compress dense RGB-D streams into discrete entities and relations that a VLM can leverage for navigation, manipulation, or question answering. However, key compression decisions—which keyframes to store, which detections to accept, which relations to represent as edges—are made without knowledge of the eventual user request. Consequently, graphs often lack precisely the objects or spatial relations required by a subsequent task, a failure mode well-documented in embodied question answering research [5, 15].

Current solutions follow two primary approaches. *Over-provisioning* retains nearly every candidate detection, incurring substantial memory and computational overhead [2]. *Offline enhancement pipelines* append heuristically chosen affordances or relations for anticipated tasks [12], but each new domain demands additional engineering effort and cannot recover information never initially detected. Neither strategy scales effectively when user requests vary widely.

Our approach: language-guided memory updates. We propose that an embodied agent should be able to *dynamically revise* its structured scene memory when reasoning reveals a gap in its knowledge. Rather than discarding an inadequate structured representation—and thereby losing its benefits—or recreating it from scratch at high computational cost, the agent should be able to modify and enhance the existing representation. This allows the agent to incorporate task-relevant information without starting over. Moreover, the agent should be able to do this through targeted perceptual queries on specific keyframes, and seamlessly integrate the newly discovered information into its scene graph.

GraphPad. We introduce **GraphPad**, a modifiable 3D scene graph memory whose content can be updated at in-

ference time through language-level commands issued by the same VLM that performs high-level reasoning. Whenever uncertainty arises or missing information is identified, the VLM can: (i) retrieve additional keyframes from its navigation log, (ii) insert previously undetected objects or spatial relationships, and (iii) annotate existing nodes with task-specific attributes. These operations are exposed as language-callable functions that the reasoning agent itself can invoke, eliminating the need for task-specific reasoning code.[1] Through successive targeted updates, the graph evolves from a coarse initial representation into a task-conditioned model containing precisely the information needed for the current query or action planning.

Contributions.

- We formulate *language-driven online editing* of structured 3D memories as a solution to the task–memory mismatch inherent in fixed scene graphs.
- We present *GraphPad*, a system in which a single VLM both identifies knowledge gaps and updates its 3D representation via language function calls during inference.
- On the OpenEQA benchmark [15], we demonstrate that GraphPad improves spatial question answering from 52.3% to 55.3% while reducing the number of processed frames from 25 to 5, outperforming both image-only and static-graph baselines without additional training.

By recasting perception as an interactive, language-mediated process rather than a static preprocessing step, GraphPad enables more efficient and effective reasoning about complex 3D environments—a critical capability for agents that must bridge language understanding with physical action.

2. Background and Related Work

Current embodied AI systems struggle with a fundamental limitation: their memory representations often lack critical information needed during task execution. This section examines key research areas relevant to this challenge.

2.1. Embodied Memory Representations

Frame-level memories store raw RGB-D streams with learned descriptors. Systems like ReMEmbR [2] and Embodied-RAG [28] index thousands of images, allowing retrieval of relevant observations during inference. While flexible, these approaches face computational overhead from processing large image collections and delegate all spatial reasoning to language models.

Semantic metric maps address these limitations by embedding visual features within geometric reconstructions. VL-Maps [10], PLA [6], and OpenScene [16] support spatial grounding but lack explicit object boundaries and relationships essential for symbolic reasoning.

3D scene graphs compress perception into discrete entities and relations. Hydra [11] pioneered real-time scene graph construction, with extensions for monocular inputs [1], multi-robot fusion [7], and open-vocabulary labeling [8]. These structures align with structured VLM prompting but remain largely static after construction.

2.2. Memory Update Mechanisms

Several existing methodologies employ updatable memories; however, they typically fall short by not dynamically refining scene graphs with new semantic information crucial for the task at hand but absent in the initial recording:

Dynamic object tracking systems like DynaMem [13], OpenIN [24], and Kimera [1] update spatial relations when objects move but rely on predefined update policies rather than reasoning-triggered modifications.

Incremental discovery methods like Moma-LLM [9] and Search3D [23] can integrate new objects but employ fixed detection heuristics rather than responding to specific reasoning gaps.

Memory selection approaches like 3D-Mem [29] and KARMA [25] retain informative views but cannot add structure absent from initial observations. Similarly, expanded-context approaches [3] can reference more images but cannot insert missing relations.

2.3. Task-Memory Alignment

Ensuring memory representations contain task-relevant information remains a central challenge:

Graph adaptation techniques attempt to tailor existing memories to specific tasks. Information-theoretic approaches [14] compress graphs while preserving task-relevant nodes. SayPlan [20] and SayNav [18] dynamically contract graphs during planning. Neither approach handles adding new information. LLM-enhanced scene graphs [12] augments an existing scene graph with task-specific affordances for the household rearrangement task. However, they rely on a complex scene graph enhancement pipeline that is domain-specific and not adaptable to diverse, unforeseen downstream tasks.

Image-level reasoning systems like Bumble [22] and TagMap [30] provide direct visual access but sacrifice explicit relational structure. Graph-based reasoning methods [28] provide structured context but treat graphs as read-only, requiring new exploration rather than memory editing.

Empirical analyses [26] confirm that missing nodes and relations—not reasoning failures—frequently cause performance errors in embodied tasks.

[1] Generic vision modules—object detector, mask extractor, depth back-projection—remain necessary but are *not* specialized for any particular downstream task.

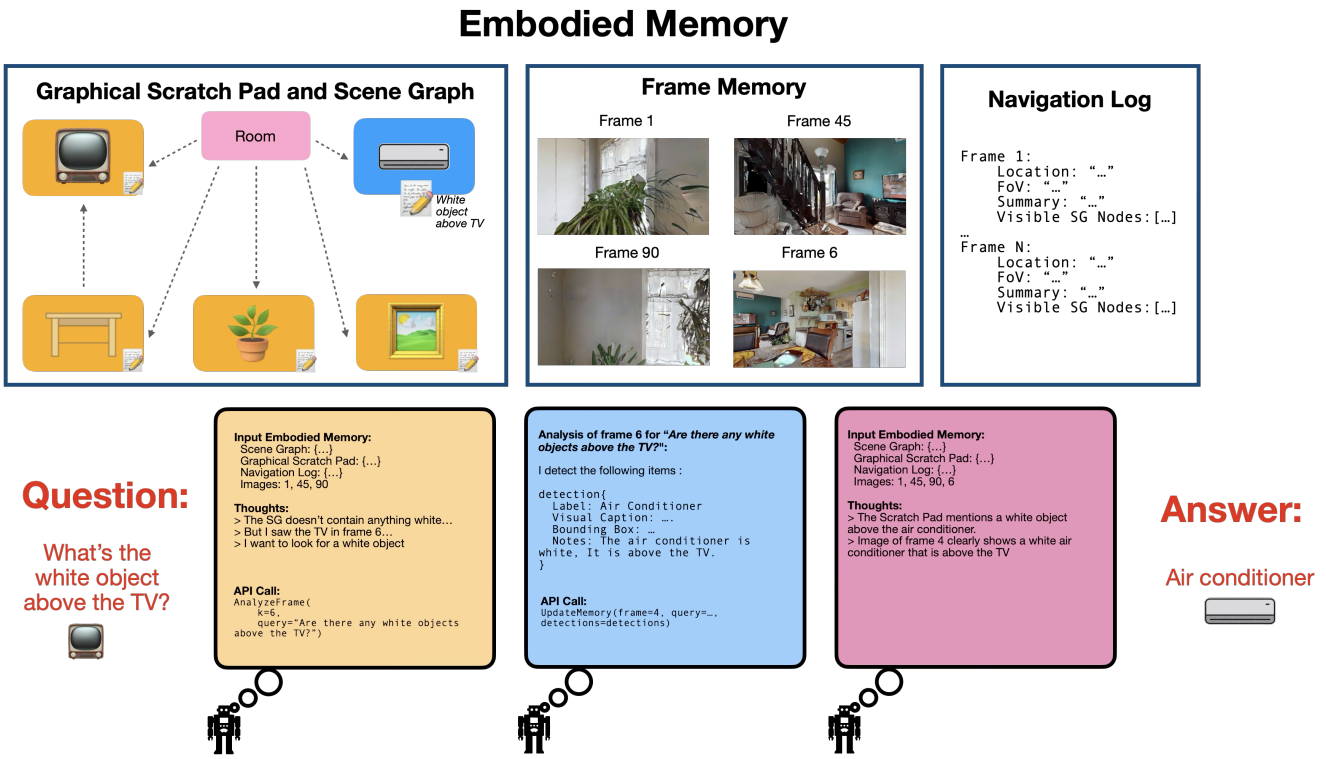

Figure 1. Overview of GraphPad for embodied question answering. The top row illustrates the Structured Scene Memory (SSM) components: Scene Graph with associated Graphical Scratch Pad, Frame Memory containing sparse keyframes, and Navigation Log indexing frame metadata. The bottom row demonstrates the inference process: given the question about a white object above the TV, the VLM first examines the initial memory state (left), identifies missing information and calls the `analyze_frame` API on a promising frame (middle), then integrates the newly detected air conditioner into the scene graph and scratch pad before providing the answer (right). This process enables dynamic, task-specific memory updates without exhaustive preprocessing.

2.4. Embodied Question Answering

Embodied Question Answering (EQA) benchmarks offer a standardized evaluation of memory adequacy. The episodic memory variant (EM-EQA) [15] provides agents with fixed observations, isolating memory representation quality from exploration.

Recent approaches to EM-EQA highlight representation completeness as a critical factor. GraphEQA [21] uses semantic graphs for viewpoint selection, while 3D-Mem [29] curates informative snapshots. Both systems nevertheless fail when queried entities are absent from their representations.

GraphPad addresses the task-memory alignment problem through targeted scene graph updates during inference. Unlike previous work, our approach enables the reasoning VLM to detect knowledge gaps and modify its own memory representation through three specific operations: frame retrieval, entity/relation insertion, and semantic annotation. This maintains the efficiency advantages of structured representations while addressing their typically static nature.

3. Methodology

GraphPad builds a *Structured Scene Memory* (SSM) from a sparse set of RGB-D keyframes and gives the vision-language model (VLM) that answers questions three callable functions to *edit* that memory during inference. We first describe the **agentic reasoning loop** (Sec. 3.1), then the four data structures that constitute the SSM (Sec. 3.2), and finally the **Modifiability APIs** (Sec. 3.3).

Throughout we follow the episodic-memory EQA protocol [15]: the agent is given every k-th RGB-D frame of a pre-recorded scan together with camera poses; no new sensing is possible after deployment.

3.1. Agentic Reasoning Loop

At test time, the VLM receives both the initial SSM and a natural language query q. The reasoning process unfolds iteratively: the VLM analyzes what information it needs to answer the question and repeatedly invokes Modifiability APIs to augment its understanding of the scene.

In each iteration, the VLM examines the current scene memory, identifies knowledge gaps relevant to the question,

and selects: (1) which API to call, (2) which frame to analyze, and (3) what specific information to seek (expressed as a natural language query). The selected API returns new objects, relations, or semantic annotations that are integrated into the scene memory. This process continues until the VLM determines it has sufficient information or reaches a maximum of m allowed API calls.

A critical constraint ensures grounded responses: the VLM must support every factual claim in its final answer with dual evidence—at least one frame from Frame Memory and corresponding notes from the Scratch-Pad. This constraint ensures that the VLM adequately utilizes its scene memory and searches the scene thoroughly prior to responding.

This approach fundamentally differs from static memory systems: instead of relying on pre-built representations to contain all necessary information, GraphPad actively identifies and fills knowledge gaps during reasoning. In our experiments, most questions are answered with just 2-3 targeted API calls, demonstrating efficient reasoning without exhaustive scene analysis.

3.2. Structured Scene Memory

The SSM contains four mutually linked structures:

Scene Graph
A directed multigraph $G = (\mathcal{N}, \mathcal{E})$ whose nodes represent object *tracks*. Each node n_i stores a point cloud P_i, pooled visual embedding V_i, pooled language embedding L_i, caption C_i, room/floor ID, and a list of keyframes in which the object is visible. Edges encode four view-invariant spatial relations critical for planning and manipulation: *on top of*, *subpart of*, *contained in*, and *attached to*.

Graphical Scratch-Pad
Mirrors $\mathcal{N}$ but adds a free-form `notes` field initialized empty; the APIs write task-specific information here during reasoning.

Frame Memory
An initial set of n_{img} evenly spaced keyframes. Additional frames requested by the APIs are appended (no eviction is used in our experiments).

Navigation Log
For each keyframe: room, textual field-of-view tag, egocentric motion label (from pose deltas), and the set of visible node IDs. The log serves as a structured index, guiding the VLM in selecting candidate frames likely to contain information about specific objects or spatial relationships.

Initial construction. For every k-th RGB image $\mathbf{I}_t$, we run a VLM detector that outputs bounding boxes and captions. Each box is passed to SAM to obtain a mask; the mask is back-projected with depth into a point cloud and voxel-downsampled to 0.02 m. Noise is removed by keeping only the largest DBSCAN cluster (default `sklearn` parameters). Visual embeddings are extracted using CLIP ViT-L/14; language embeddings come from BGE [27].

Track association. A new detection D_i is matched to an existing track T_j when the vote

$$S_{ij} = \mathbf{1}[V_i \cdot V_j > 0.7] + \mathbf{1}[L_i \cdot L_j > 0.8] + \mathbf{1}[G_{ij} > 0.4] \quad (1)$$

exceeds 2, where G_{ij} is the fraction of points in D_i within δ_g of T_j (we use $\delta_g = 5$ cm). This voting scheme requires at least two of the following conditions to hold with sufficient confidence: visual feature similarity, caption semantic similarity, or spatial overlap. Visual and language embeddings of the matched track are updated by an exponential moving average with $\alpha = 0.5$; unmatched detections start new tracks.

Edge discovery. Every three frames we prompt the VLM with the current frame plus the JSON list of visible $\langle$*bbox*, caption$\rangle$ pairs; the model predicts all pairwise relations among the four relation types. Each predicted edge is stored with a subject-ID, object-ID, relation label, and the VLM's free-form justification string.

Caption consolidation. Accumulated captions on a track are periodically compressed by prompting the VLM with the list and asking for a single sentence that faithfully paraphrases all entries.

Room/floor labels. We adopt the HOV-SG pipeline [26]: floors from height-histogram modes and rooms via watershed segmentation on wall skeletons; room labels derive from CLIP similarity to a fixed set of class names.

When the SSM is passed to the VLM for reasoning, the Scene Graph, Scratch-Pad, and Navigation Log are serialized to JSON, while Frame Memory is supplied as interleaved images with their Frame IDs.

3.3. Modifiability APIs

All three APIs receive a *Frame ID* and a natural-language *query*. Each returns a JSON patch containing new nodes, edges, or scratch-pad notes plus evidence pointers that are incorporated into the SSM.

`find_objects`
Detects previously unseen instances relevant to *query* in the specified frame and fuses them into G. The function leverages the VLM's bounding box detection capabilities to identify query-relevant objects and generate corresponding query-relevant notes. These detections are incorporated into the scene graph by associating them with existing tracks or creating new ones. The

query-relevant notes are used to update the corresponding scratch-pad entries.

`analyze_objects`
For each node in the user-supplied list that is visible in the frame, the VLM analyzes its appearance and answers *query*, storing the result in the node's `notes`. The function employs the VLM to examine each visible node's bounding box and generate descriptive notes pertaining to the query. These notes are stored in the corresponding nodes' scratch-pad entries. If specified nodes are not visible in the frame, the function defaults to `find_objects`.

`analyze_frame`
A frame-level variant that jointly discovers undetected objects and annotates existing ones with respect to *query*. This consolidated approach can both identify new perceptual elements and enrich the semantic understanding of known objects in a single operation.

These APIs enable a critical capability: the reasoning agent itself can identify and remedy gaps in its scene representation during inference. Rather than preprocessing exhaustively for anticipated questions, GraphPad builds a minimal initial representation and lets task requirements guide targeted perceptual refinement. This approach aligns with real-world robotic scenarios where complete scene understanding is computationally intractable, but targeted perception can efficiently support specific goals.

By invoking these functions, the agent patches omissions in its memory in real time, yielding a task-conditioned graph that improves both answer accuracy and confidence without requiring rescanning of the physical scene.

4. Results

We evaluate GraphPad on the OpenEQA benchmark [15] to assess how language-guided scene graph updates affect spatial reasoning performance. Our experiments use Gemini 2.0 Flash [17] as both the reasoning agent and detector. All evaluations use the episodic-memory variant of OpenEQA, where systems receive fixed keyframes from 3D scene scans (HM3D [19] and ScanNet [4]).

4.1. Baseline Comparison on OpenEQA

Table 1 presents GraphPad's performance against established baselines. Using an initial frame memory size of $n_{img} = 5$, search depth of $m = 20$, and Frame-Level API, GraphPad achieves 55.3% accuracy. This represents a 3.0 percentage point increase over using the same VLM (Gemini 2.0 Flash) with image-only input (52.3%). GraphPad processes 5 initial frames compared to the 25 frames (every k-th frame with $k = 5$ for HM3D and $k = 20$ for ScanNet) used in the image-only baseline.

Table 1. OpenEQA performance comparison across methods

#	Method	Accuracy (%)
Blind LLMs		
1	GPT-4	33.5
Socratic LLMs w/ Frame Captions		
2	GPT-4 w/ LLaVA-1.5	43.6
Socratic LLMs w/ Scene-Graph Captions		
3	GPT-4 w/ CG	36.5
4	GPT-4 w/ SVM	38.9
Multi-Frame VLMs		
5	GPT-4V	55.3
6	Gemini-2.0 Flash	52.3
7	3D-Mem (w/ GPT4V)	**57.2**
Human Agent		
8	Human	86.8
Our Results		
9	GraphPad (w/ Gemini 2.0 Flash)	55.3

Table 2. Ablation study of system components

Method	Accuracy (%)
Frame Memory	32.9
Frame Memory + Scene Graph	34.6
Frame Memory + Navigation Log	42.5
Frame Memory + SG + Navigation Log	46.9
Frame Memory + Navigation Log + Image API	45.1
Frame Memory + SG + Navigation Log + Node-level API	47.1
Frame Memory + SG + Navigation Log + Frame-level API	**50.5**

GraphPad scores higher than static scene graph methods (CG: 36.5%, SVM: 38.9%), suggesting potential benefits of dynamic scene graph updates during inference. The system performs identically to GPT-4V (55.3%) despite using a different base VLM. 3D-Mem [29] achieves 57.2% accuracy, outperforming our approach by 1.9 percentage points, though it uses GPT-4V rather than Gemini.

The comparison suggests that structured memory with targeted updates can help bridge the performance gap between different vision-language models while potentially reducing the number of frames that need to be processed.

4.2. Component Analysis

To understand the contribution of different GraphPad components, we conducted an ablation study (Table 2) using a subset of 184 OpenEQA questions. Starting with just four raw frames (32.9% accuracy), we progressively added components to measure their individual impact.

The Navigation Log provides the largest individual improvement (+9.6 percentage points over frame-only), suggesting that structured information about frame contents helps guide the VLM's attention. Adding a static scene

Table 3. Performance as a function of search depth

Search Depth	Accuracy (%)
0	46.9
1	45.8
2	46.3
3	48.1
4	45.9
5	49.7
20	**51.2**

Table 4. Performance as a function of initial frame count

Initial Frames	Accuracy (%)
2	48.2
3	47.4
4	49.9
5	50.3
6	**50.5**

graph yields a modest improvement (+1.7 percentage points over frame-only), while the Modifiability APIs add another 3.6 percentage points over the static scene representation.

Frame-level APIs (50.5%) outperformed node-level variants (47.1%). We observed that when using node-level APIs, the VLM tended to rely more on `find_objects` than `analyze_objects`, often using the former in ways similar to `analyze_frame`.

4.3. Effect of Search Depth and Frame Count

We examined how search depth (m) affects accuracy using the OpenEQA184 subset with $n_{img} = 4$ initial frames and the Frame-Level API. Table 3 shows that accuracy generally increases with search depth, reaching 51.2% at $m = 20$. Interestingly, limited search ($m = 1$ or $m = 2$) sometimes performs worse than no search ($m = 0$), possibly because preliminary updates without follow-up refinement can introduce misleading information.

The initial number of frames in Frame Memory also influences performance (Table 4). Accuracy improves as more frames are included, though gains diminish beyond 5 frames. We observed that with fewer initial frames, the model made more API calls on average, potentially compensating for the limited initial context.

4.4. Performance by Question Category

Breaking down performance by question category (Table 5) reveals variation in how GraphPad (with $k = 5$) compares to using Gemini-2.0 with only images (every 5th frame in the scene).

GraphPad shows larger improvements in attribute recognition (+20.3 percentage points), functional reasoning (+5.7 percentage points), and object state recognition (+3.1 percentage points). These categories often require detailed analysis of specific object properties. The image-only baseline performs better in object recognition (-3.8 percentage points), spatial understanding (-4.7 percentage points), and object localization (-3.0 percentage points), which may rely more on raw visual processing capabilities.

Table 5. Performance by question category: GraphPad vs. Gemini 2.0 with images only

Category	GraphPad (%)	Gemini (%)
Attribute Recognition	**66.8**	46.5
Object State Recognition	**69.6**	66.5
Functional Reasoning	**59.2**	53.5
World Knowledge	**55.9**	52.0
Object Recognition	58.4	**62.2**
Spatial Understanding	47.7	**52.4**
Object Localization	31.3	**34.3**

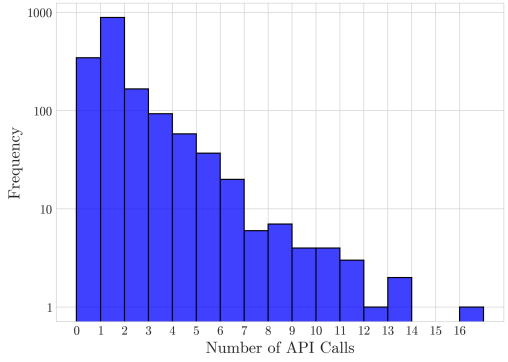

Figure 2. Distribution of API calls per query on OpenEQA

4.5. API Call Distribution

Analysis of GraphPad's API usage (Fig. 2) shows that 95% of questions are answered with 5 or fewer API calls, with an average of 1.9 calls per question. The distribution indicates that in most cases, the system requires relatively few updates to answer questions, though we observe a long tail of more complex queries requiring additional refinement steps.

The peak at zero calls represents questions that could be answered using only the initial scene representation without any additional API calls, suggesting that for some question types, the initial structured memory provides sufficient context.

5. Limitations

While GraphPad demonstrates the value of editable 3D scene representations, several important limitations affect its current implementation:

- **Error propagation in detection.** Our current implementation lacks a verification mechanism for object detection

quality. When the VLM misidentifies an object or relation, this error becomes part of the scene graph and can negatively affect downstream reasoning.

- **API design constraints.** The three operations (find, analyze objects, analyze frame) represent our initial attempt at a minimal API set, but we have not rigorously evaluated whether this is optimal. Different reasoning tasks might benefit from specialized APIs not explored in this work.
- **Computational overhead.** Each API call requires a full VLM inference pass, adding significant latency (typically 2-3 seconds per call in our implementation). This latency currently limits GraphPad's applicability to real-time systems.
- **Domain generalization.** Our APIs were designed specifically for question answering about static scenes. We have not tested their effectiveness for other domains like manipulation planning or navigation in dynamic environments.
- **Scalability limits.** As scene graphs grow larger, both the prompt size and reasoning complexity increase. Our experiments were limited to medium-sized home environments; performance in larger spaces remains untested.

6. Conclusion

We presented GraphPad, a system that enables VLMs to update 3D scene graphs during inference through language-callable functions. On OpenEQA, GraphPad improved accuracy by 3.0 percentage points over an image-only baseline using the same VLM while requiring fewer input frames.

Our experiments suggest several insights for 3D language-vision systems. First, static structured representations appear to benefit from targeted, task-specific refinement. Second, language-guided perception may offer a middle ground between exhaustive preprocessing and purely reactive vision. Third, the efficiency of GraphPad (averaging under 2 API calls per question) indicates that targeted scene exploration can be a practical strategy.

For future 3D vision-language-action systems, these results suggest investigating how reasoning agents might direct their own perception in service of task goals. Extending GraphPad's approach to manipulation planning, navigation, and dynamic scenes could help bridge the gap between language understanding and effective action in 3D environments.

References

[1] M. Abate N. Hughes Y. Chang J. Shi A. Gupta L. Carlone A. Rosinol, A. Violette. Kimera: from SLAM to spatial perception with 3D dynamic scene graphs. In *arxiv*, 2021. 2

[2] Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. *arXiv preprint arXiv:2409.13682*, 2024. 1, 2

[3] Hao-Tien Lewis Chiang, Zhuo Xu, Zipeng Fu, Mithun George Jacob, Tingnan Zhang, Tsang-Wei Edward Lee, Wenhao Yu, Connor Schenck, David Rendleman, Dhruv Shah, Fei Xia, Jasmine Hsu, Jonathan Hoech, Pete Florence, Sean Kirmani, Sumeet Singh, Vikas Sindhwani, Carolina Parada, Chelsea Finn, Peng Xu, Sergey Levine, and Jie Tan. Mobility vla: Multimodal instruction navigation with long-context vlms and topological graphs. *arXiv preprint arXiv:2407.07775*, 2024. 2

[4] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 5

[5] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[6] Runyu Ding, Jihan Yang, Chuhui Xue, Wenqing Zhang, Song Bai, and Xiaojuan Qi. Pla: Language-driven open-vocabulary 3d scene understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7010–7019, 2023. 2

[7] Elias Greve, Martin Büchner, Niclas Vödisch, Wolfram Burgard, and Abhinav Valada. Collaborative dynamic 3d scene graphs for automated driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11118–11124, 2024. 2

[8] Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028. IEEE, 2024. 2

[9] Daniel Honerkamp, Martin Büchner, Fabien Despinoy, Tim Welschehold, and Abhinav Valada. Language-grounded dynamic scene graphs for interactive object search with mobile manipulation. *IEEE Robotics and Automation Letters*, 2024. 2

[10] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023. 2

[11] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization. *arXiv preprint arXiv:2201.13360*, 2022. 1, 2

[12] Wenhao Li, Zhiyuan Yu, Qijin She, Zhinan Yu, Yuqing Lan, Chenyang Zhu, Ruizhen Hu, and Kai Xu. Llm-enhanced scene graph learning for household rearrangement. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 1, 2

[13] Peiqi Liu, Zhanqiu Guo, Mohit Warke, Soumith Chintala, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel

Pinto. Dynamem: Online dynamic spatio-semantic memory for open world mobile manipulation. *arXiv preprint arXiv:2411.04999*, 2024. 2

[14] Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clio: Real-time task-driven open-set 3d scene graphs. *arXiv preprint arXiv:2404.13696*, 2024. 2

[15] Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16488–16498, 2024. 1, 2, 3, 5

[16] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023. 2

[17] Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. Introducing Gemini 2.0: our new AI model for the agentic era, 2024. 5

[18] Abhinav Rajvanshi, Karan Sikka, Xiao Lin, Bhoram Lee, Han pang Chiu, and Alvaro Velasquez. Saynav: Grounding large language models for dynamic planning to navigation in new environments. In *34th International Conference on Automated Planning and Scheduling*, 2024. 2

[19] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. 5

[20] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. In *7th Annual Conference on Robot Learning*, 2023. 2

[21] Saumya Saxena, Blake Buchanan, Chris Paxton, Bingqing Chen, Narunas Vaskevicius, Luigi Palmieri, Jonathan Francis, and Oliver Kroemer. Grapheqa: Using 3d semantic scene graphs for real-time embodied question answering. *arXiv preprint arXiv:2412.14480*, 2024. 3

[22] Rutav Shah, Albert Yu, Yifeng Zhu, Yuke Zhu, and Roberto Martín-Martín. Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation. *arXiv preprint arXiv:2410.06237*, 2024. 2

[23] Ayca Takmaz, Alexandros Delitzas, Robert W Sumner, Francis Engelmann, Johanna Wald, and Federico Tombari. Search3d: Hierarchical open-vocabulary 3d segmentation. *arXiv preprint arXiv:2409.18431*, 2024. 2

[24] Yujie Tang, Meiling Wang, Yinan Deng, Zibo Zheng, Jingchuan Deng, and Yufeng Yue. Openin: Open-vocabulary instance-oriented navigation in dynamic domestic environments. *arXiv preprint arXiv:2501.04279*, 2025. 2

[25] Zixuan Wang, Bo Yu, Junzhe Zhao, Wenhao Sun, Sai Hou, Shuai Liang, Xing Hu, Yinhe Han, and Yiming Gan. Karma: Augmenting embodied ai agents with long-and-short term memory systems. *arXiv preprint arXiv:2409.14908*, 2024. 2

[26] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. 2, 4

[27] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings, 2024. 4

[28] Quanting Xie, So Yeon Min, Tianyi Zhang, Kedi Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-rag: General non-parametric embodied memory for retrieval and generation. *arXiv preprint arXiv:2409.18313*, 2024. 2

[29] Yuncong Yang, Han Yang, Jiachen Zhou, Peihao Chen, Hongxin Zhang, Yilun Du, and Chuang Gan. 3d-mem: 3d scene memory for embodied exploration and reasoning, 2024. 2, 3, 5

[30] Mike Zhang, Kaixian Qu, Vaishakh Patil, Cesar Cadena, and Marco Hutter. Tag map: A text-based map for spatial reasoning and navigation with large language models. *arXiv preprint arXiv:2409.15451*, 2024. 2